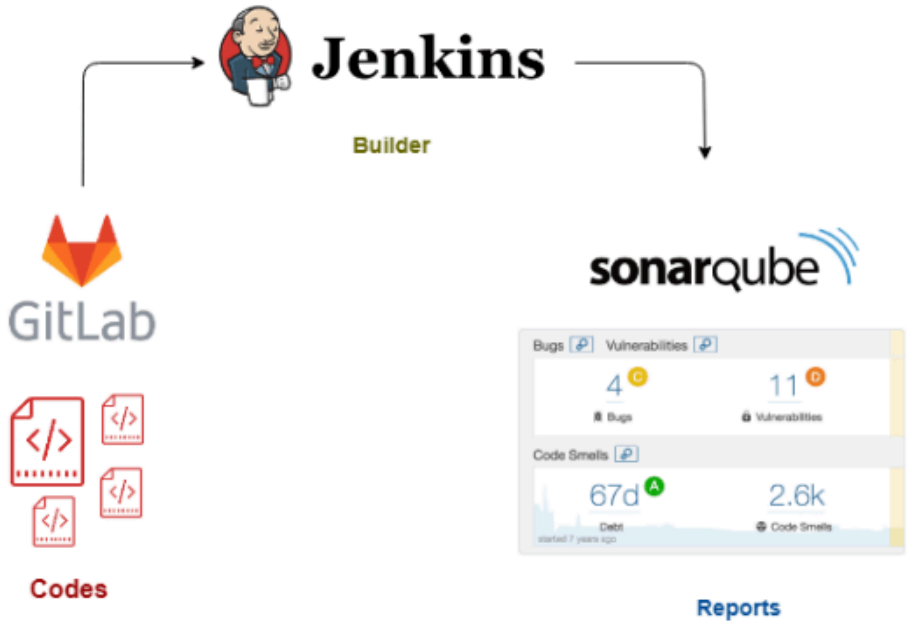
Sonarqube Integration with Jenkins using CI/CD Pipeline



Prerequisites:

- OS - Ubuntu-20.04
- Container Engine - Docker & Docker Compose
- CI / CD Tool - Jenkins
- Jenkins Plugins - SonarQube Scanner GitLab Plugin

Step # 1 Set VM parameters

For SonarQube, you need to set the recommended values as a root user on the host machine:

```
sysctl -w vm.max_map_count=262144
sysctl -w fs.file-max=65536
ulimit -n 65536
ulimit -u 4096
```

Step # 2 Deployment of SonarQube with Docker

https://docs.sonarqube.org/latest/setup-and-upgrade/install-the-server/

If you want to deploy below docker-compose with non sudo user; it must have sudo privilege

```
mkdir sonar && cd sonar
nano docker-compose.yml
```

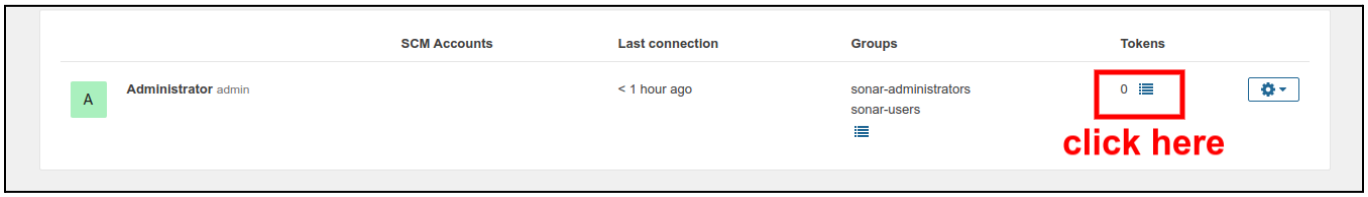
```
version: "3"

services:
  sonarqube:
    image: sonarqube:community
    depends_on:
      - db
    environment:
      SONAR_JDBC_URL: jdbc:postgresql://db:5432/sonar
      SONAR_JDBC_USERNAME: sonar
      SONAR_JDBC_PASSWORD: sonar
    volumes:
      - ./sonarqube/data:/opt/sonarqube/data
      - ./sonarqube/extensions:/opt/sonarqube/extensions
      - ./sonarqube/logs:/opt/sonarqube/logs
    ports:
      - "9000:9000"
  db:
    image: postgres:12
    environment:
      POSTGRES_USER: sonar
      POSTGRES_PASSWORD: sonar
    volumes:
      - ./sonarqube/postgresql:/var/lib/postgresql
      - ./sonarqube/postgresql/postgresql_data:/var/lib/postgresql/data
```

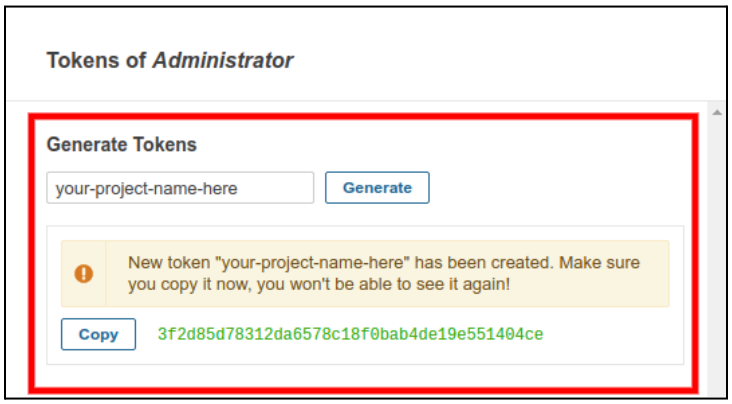
Run the following command to launch the stack:

```
docker-compose up -d
```

Step # 3 SonarQube Configuration

To connect it with Jenkins, you need to generate the token to access the SonarQube instance.

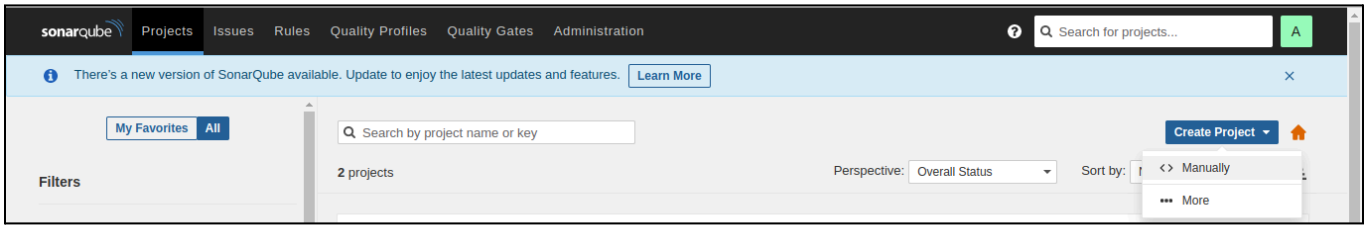




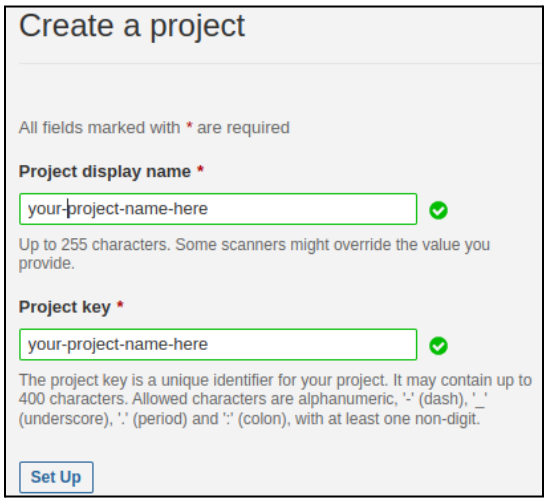
NOTE: Copy the newly generated token and save it somewhere safe, as you won't be able to view/copy the generated token again. It will be used in Jenkins plugin

Step # 4 Setup project in Sonarqube

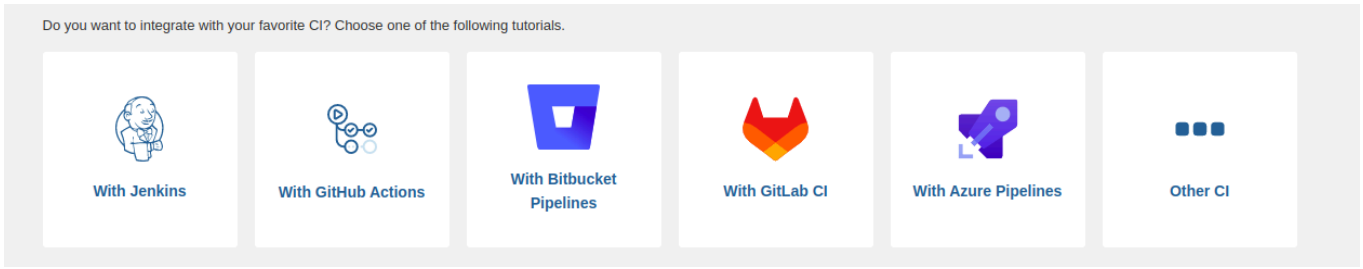
Projects > Create Project > Manually



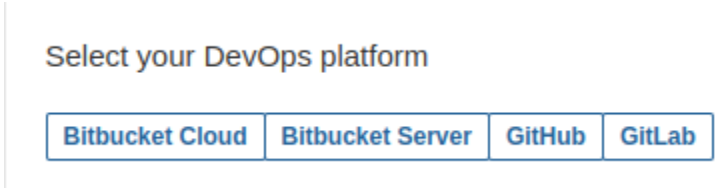
Enter project name same as Jenkins pipeline



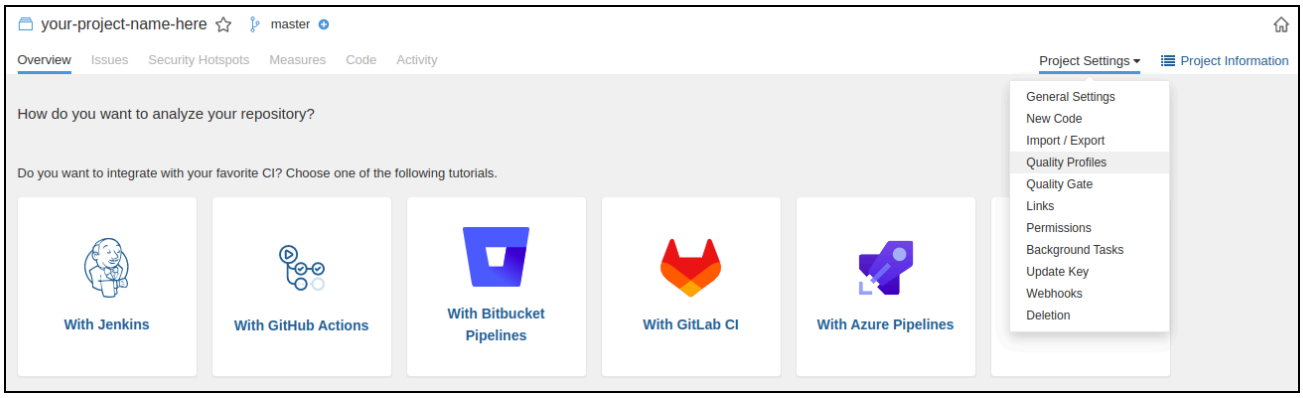
Select **With Jenkins**



Select **Gitlab**

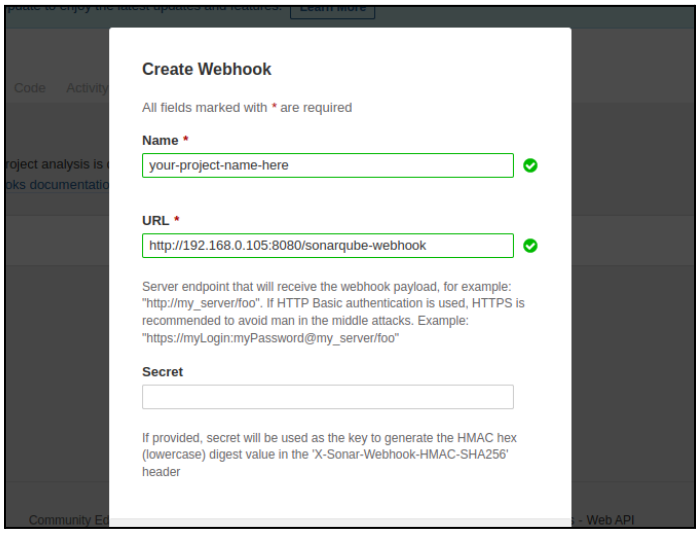


Set Quality Profiles || Quality Gate as per your requirement



Create a webhook for your project with the same project name; **it will send scan result analysis to Jenkins**.

Enter the project name same as above in the name field
https://${jenkins_domain}/sonarqube-webhook/



Step # 5 Deploy Sonar-Scanner in Jenkins server

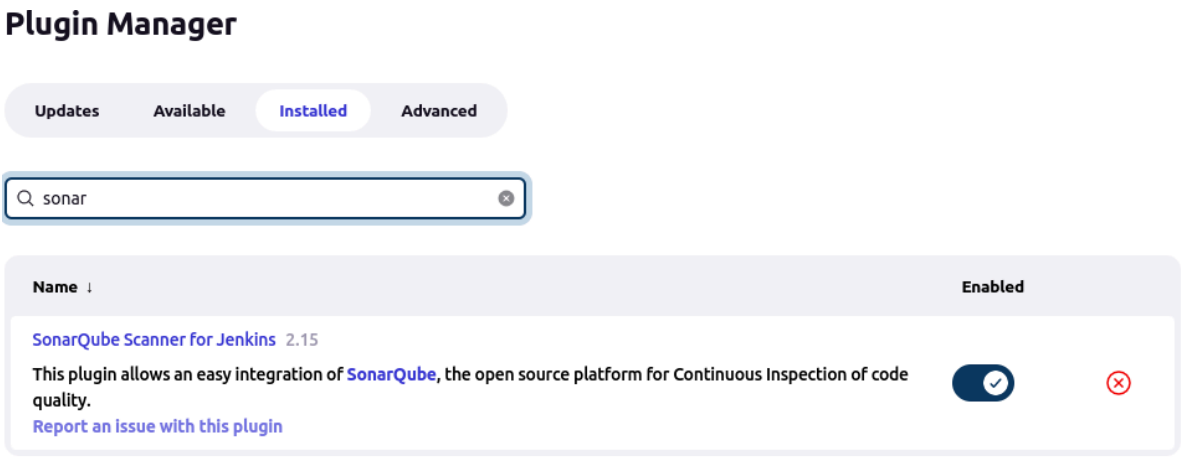
Login to Jenkins machine

```
cd /var/lib/jenkins/
wget
https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-4.8.0.2856-linux.zip

unzip sonar-scanner-cli-4.8.0.2856-linux.zip
cd sonar-scanner-4.8.0.2856-linux
```

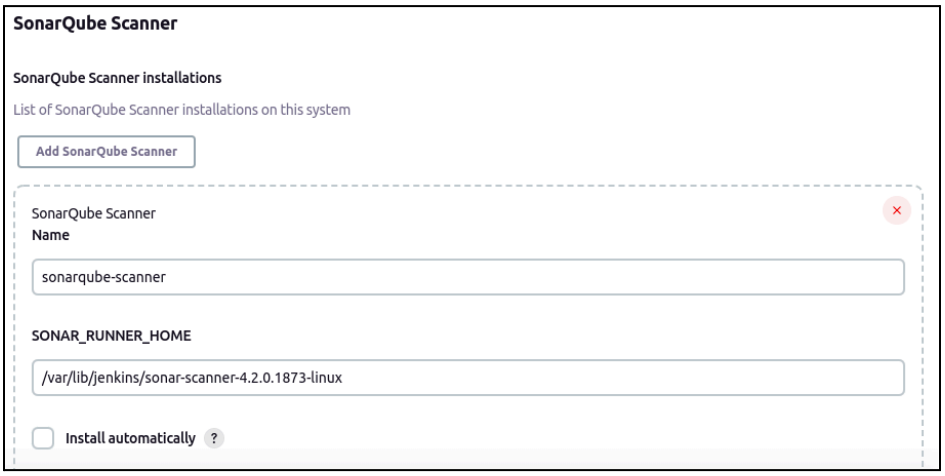
```
pwd  //(copy the path, it will use in jenkins sonar-scanner plugin configuration)
```

Step # 6 Configure the Sonar-Scanner plugin on the Jenkins server

Install Jenkins Plugin through Plugin Manager



Go to Manage Jenkins page and select Global Tool Configuration.
Enter the name “sonarqube-scanner”, it will be used in the pipeline. Also, paste the path copied in Step 5.

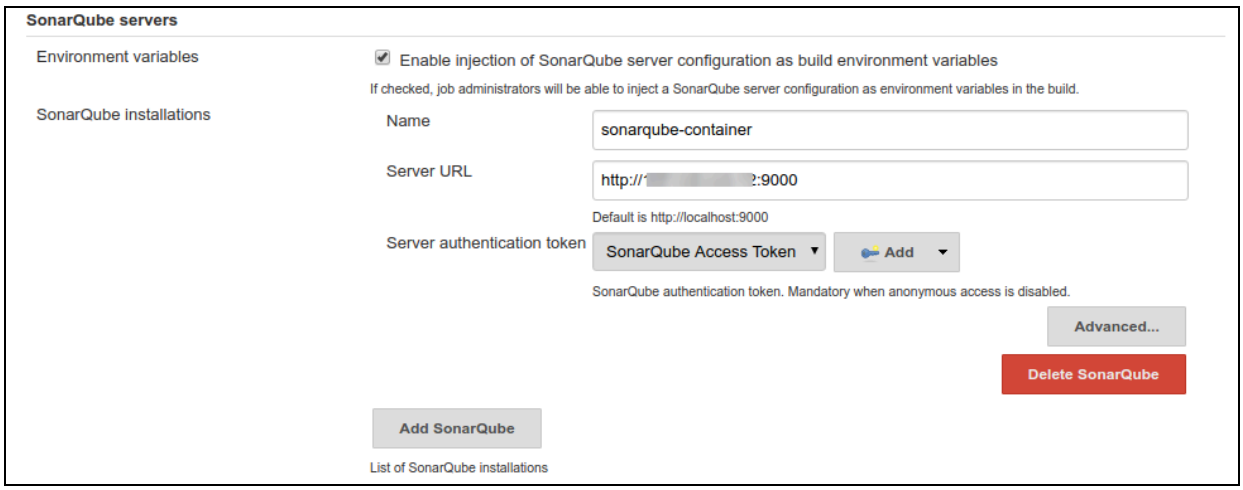


Create credentials for the Sonarqube server in Jenkins server

1. Jenkins Dashboard, navigate to Credentials > System from the left navigation.
2. Click the Global credentials (unrestricted) link in the System table.
3. Click Add credentials in the left navigation and add the following information:
4. Kind: Secret Text
5. Scope: Global
6. Secret: paste the Token that was copied at step 3.

Enter configuration name and copy the name which will use in pipeline

Go to Dashboard > Manage Jenkins > Configure System



Step # 7 Add Creating Jenkins Pipeline

==========

```
pipeline {
agent any
     environment{
          credentialsId = "gitlab-idpass"
          branch= "main"
          sonarqube_project_name=”testproject1”
}
stages {
 stage("Code Checkout from GitLab") {
  steps {
    git branch: branch,
    credentialsId: credentialsId,
    url: repo_url
  }
 }
```

```
 stage('Code Quality Check via SonarQube') {
  steps {
    script {
       def scannerHome = tool 'sonarqube-scanner';
           withSonarQubeEnv("sonarqube-container") {
           sh "${scannerHome}/bin/sonar-scanner
-Dsonar.projectKey=sonarqube_project_name”
               }
           }
       }
   }

 stage("Quality Gate") {
  steps {
        timeout(time: 10, unit: 'MINS') {
// Parameter indicates whether to set pipeline to UNSTABLE if Quality Gate
fails
// true = set pipeline to UNSTABLE, false = don't
        waitForQualityGate abortPipeline: true
        }
    }
  }


 }
}
```

NOTE:

TO CHANGE SONARQUBE BRANCH

https://sonarqube.server.url/project/branches?id=<your project name>

- **Click on setting button**
- **Edit the branch name**

References:

- https://dzone.com/articles/jenkins-pipeline-with-sonarqube-and-gitlab
- https://docs.sonarqube.org/latest/analysis/scan/sonarscanner-for-jenkins/
- https://linuxhandbook.com/ci-with-gitlab-jenkins-and-sonarqube/